What is Program Synthesis?

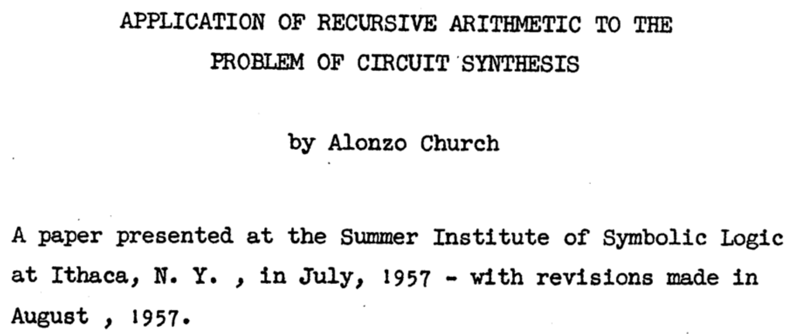
$$\text{Specification (Dream)} \longrightarrow \text{Program}$$

Holy grail of computer science

Some history

1957 | 1980 | 1982 | 2011 | 2022

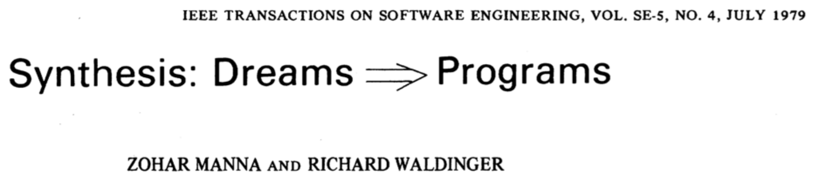
Alonzo Church


APPLICATION OF RECURSIVE ARITHMETIC TO THE PROBLEM OF CIRCUIT SYNTHESIS

by Alonzo Church

A paper presented at the Summer Institute of Symbolic Logic at Ithaca, N. Y. , in July, 1957 - with revisions made in August , 1957.

Zohar Manna Richard Waldinger


IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-5, NO. 4, JULY 1979

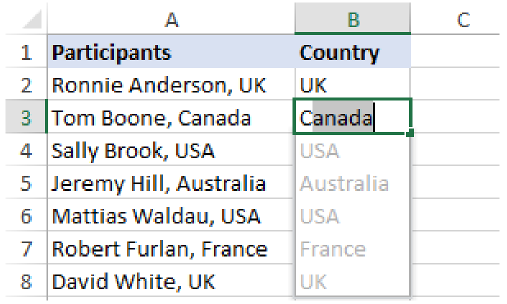
Synthesis: Dreams $\Longrightarrow$ Programs

ZOHAR MANNA AND RICHARD WALDINGER

Alan Perlis

"When someone says 'I want a programming language in which I need only say what I wish done,' give him a lollipop."

Sumit Gulwani



	A	B	C
1	**Participants**	**Country**	
2	Ronnie Anderson, UK	UK	
3	Tom Boone, Canada	Canada	
4	Sally Brook, USA	USA	
5	Jeremy Hill, Australia	Australia	
6	Mattias Waldau, USA	USA	
7	Robert Furlan, France	France	
8	David White, UK	UK	

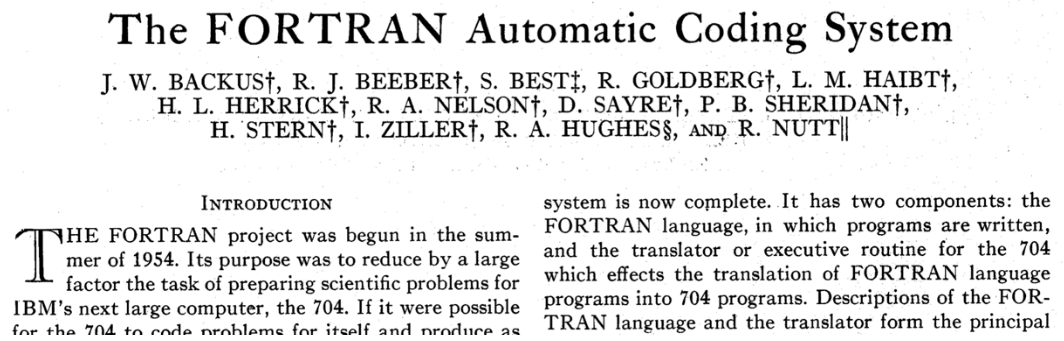
Synthesis vs. Compilation

Theoretically, indistinguishable

- Compilers were touted as program synthesizers
- Program synthesizers will be super compilers
- *“A synthesizer is a compiler that doesn’t work.”* – Eran Yahav
- “AI is whatever hasn’t been done yet.” – Tesler’s Theorem

Practically, the line is blurry

- Compilers translate, synthesizers search.
- *Superoptimization*: synthesizing an optimal sequence of instructions [1]
- *Autotuning:* searching the space of optimizations [2]



The FORTRAN Automatic Coding System

J. W. BACKUS†, R. J. BEEBER†, S. BEST‡, R. GOLDBERG†, L. M. HAIBT†, H. L. HERRICK†, R. A. NELSON†, D. SAYRE†, P. B. SHERIDAN†, H. STERN†, I. ZILLER†, R. A. HUGHES§, AND R. NUTT||

INTRODUCTION

THE FORTRAN project was begun in the summer of 1954. Its purpose was to reduce by a large factor the task of preparing scientific problems for IBM's next large computer, the 704. If it were possible for the 704 to code problems for itself and produce as

system is now complete. It has two components: the FORTRAN language, in which programs are written, and the translator or executive routine for the 704 which effects the translation of FORTRAN language programs into 704 programs. Descriptions of the FORTRAN language and the translator form the principal

[1] Massalin. Superoptimizer – A Look at the smallest Program. ASPLOS’87.

[2] Ansel et al. PetaBricks: A Language and Compiler for Algorithmic Choice. PLDI’09

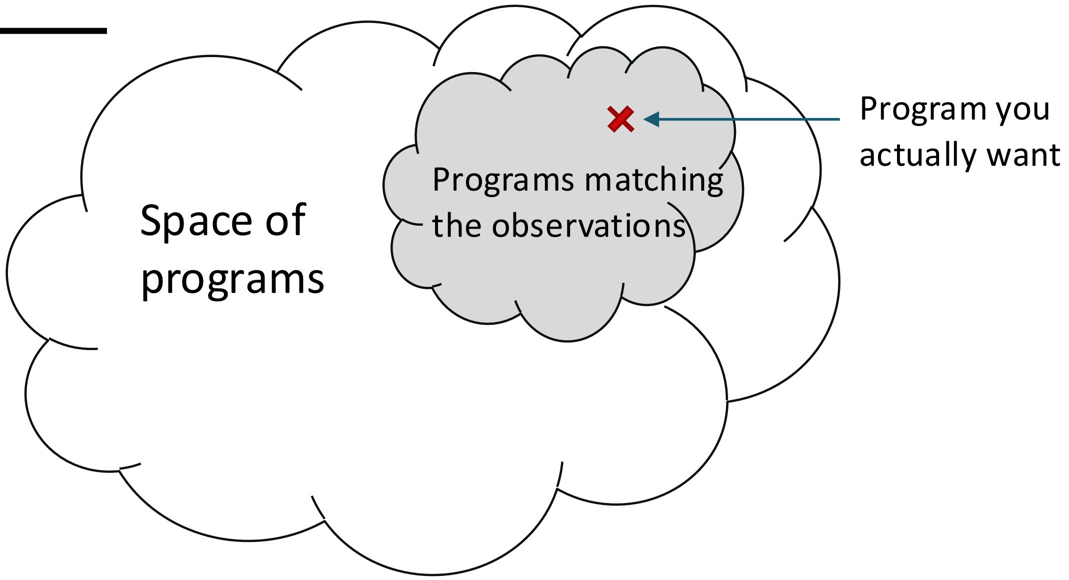
Synthesis vs. Machine Learning

Similarity between ML and synthesis

- ML synthesizes inscrutable programs (e.g., neural nets) from large, noisy sets of samples *(interpretability, robustness, overtraining)*
- (Inductive) synthesis learns natural, discrete programs from small, precise examples *(user interaction)*
- *Neurosymbolic programs [1]*

Different focuses

- ML focuses on the second question (avoiding over/under-fitting)
- Modern synthesis supports more flexible program spaces and focuses on the first question

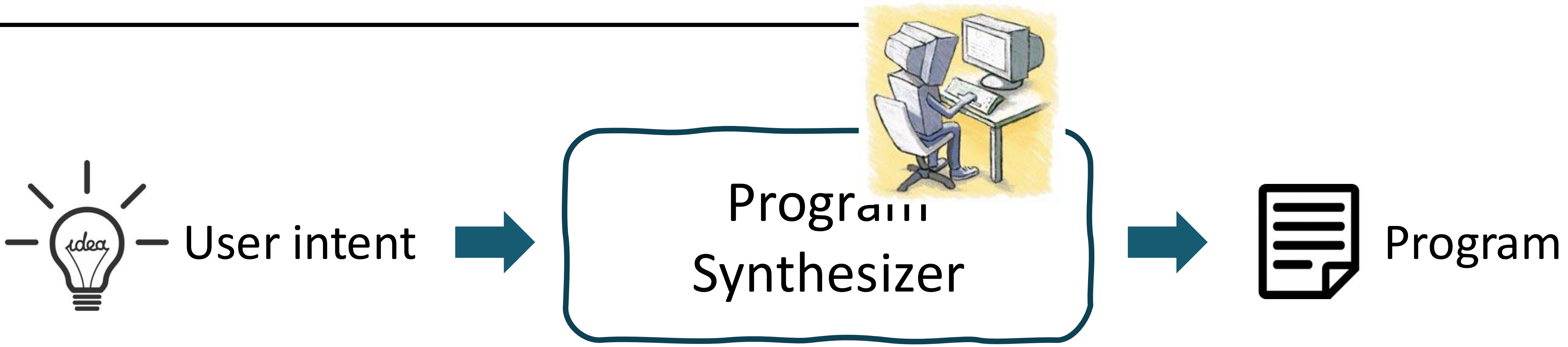


(1) How do you find a program that matches the observations?

(2) How do you know it is the program you are looking for?

[1] http://www.neurosymbolic.org/

Challenges



Intention

E.g., Programming By Example (PBE), program sketching

Invention

E.g., Counterexample-Guided Inductive Synthesis (CEGIS)

Adaptation

E.g., Superoptimization

The Three Pillars of Machine Programming. [Gottschlich et al., MAPL 2018]

Game Plan

Intention: How to describe a problem?

Multimodal Specifications

- Mathematical Logic
- Examples
- Natural Languages

Invention: How to produce a program?

Synthesis Algorithms

- Deduction
- Enumeration
- Neural Approaches

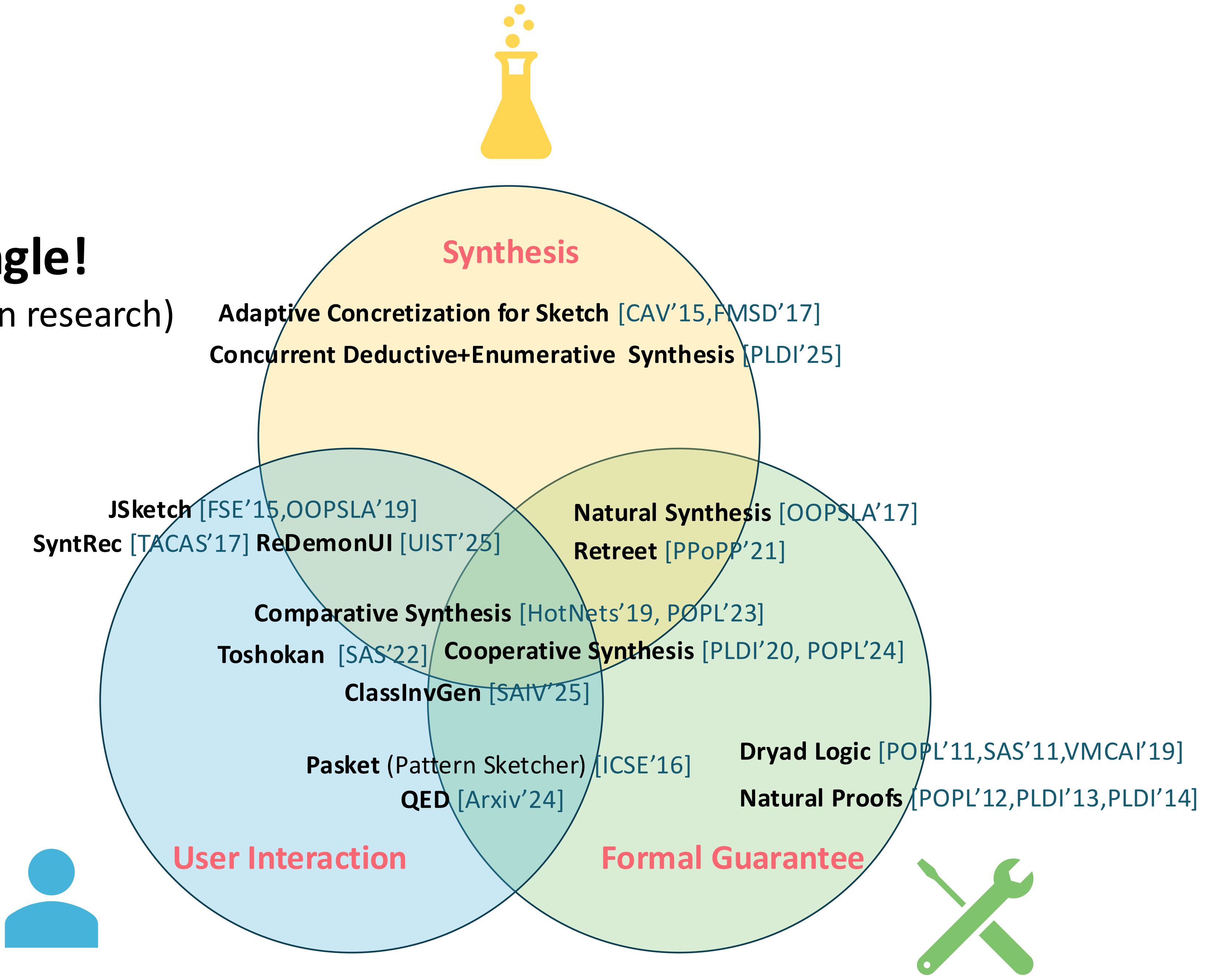
Adaptation: How to check if the produced program is the desired one?

Interdisciplinary

- Optimization
- Human-Computer Interaction
- Generative AI

They all intermingle!

(as illustrated by my own research)


Synthesis
Adaptive Concretization for Sketch [CAV'15,FMSD'17]
Concurrent Deductive+Enumerative Synthesis [PLDI'25]
JSketch [FSE'15,OOPSLA'19]
SyntRec [TACAS'17] ReDemonUI [UIST'25]
Natural Synthesis [OOPSLA'17]
Retreet [PPoPP'21]
Comparative Synthesis [HotNets'19, POPL'23]
Toshokan [SAS'22]
Cooperative Synthesis [PLDI'20, POPL'24]
ClassInvGen [SAIV'25]
Pasket (Pattern Sketcher) [ICSE'16]
QED [Arxiv'24]
Dryad Logic [POPL'11,SAS'11,VMCAI'19]
Natural Proofs [POPL'12,PLDI'13,PLDI'14]
User Interaction
Formal Guarantee